CS2100: Computer Organisation
Tutorial #2: Boolean Algebra and Logic Gates
(30 January – 3 February 2012)

[This document is available on course website http://www.comp.nus.edu.sg/~cs2100/3_ca/tutorials.html]

Please prepare yourself well for tutorials and participate actively in class. Tutorials are not tests, so do discuss among yourselves, before and after tutorials, and post your answers and queries in the IVLE forum. We usually do not provide answers to tutorial questions.

To save time and to promote participation in the forum, Q1 and Q2 will NOT be discussed in tutorial. Please discuss them on the forum.

1. One common mistake students made is the following:

   $A \cdot B + A' \cdot B' = 1$ … (equation 1)

   This seems to be erroneously "derived" from the following rule:

   $X + X' = 1$

   How do you <u>prove that equation 1 is incorrect</u>?

   Is the following equation correct?

   $A \cdot B + (A \cdot B)' = 1$ … (equation 2)

2. Given the following two 3-variable Boolean functions:

   $F(A,B,C) = \Sigma\, m(0, 2, 4, 6, 7)$
   $G(A,B,C) = \Sigma\, m(1, 2, 3, 6)$

   (a) Write the product-of-maxterms expressions in ΠM notation for F and G.

   (b) If $Y = F + G$, write the sum-of-minterms expressions in Σm notation for Y. (Can you generalise this for any arbitrary Boolean functions F and G?)

   (c) If $Z = F \cdot G$, write the sum-of-minterms expressions in Σm notation for Z. (Can you generalise this for any arbitrary Boolean functions F and G?)

3. Using Boolean algebra, simplify each of the following expressions into minimal sum-of-products (SOP) expressions. Indicate the laws/theorems used.

   (a) $(b' + a' \cdot c)' + (a + c)'$

   (b) $p \cdot r + p \cdot q \cdot r + r' \cdot (p' \cdot q' + p' \cdot q + p \cdot q' + p \cdot \text{q})$

   (c) $g(A,B,C,D) = \Pi\, M(2,4,6)$

   For (c), it is simpler to start with the given product-of-sums expression, rather than to convert it into sum-of-products expression first.

4. Express the functions in question 3 in the Σm-notation.

5. Design a divide-by-3 circuit as follows: the input is a 4-bit unsigned binary number *ABCD*, and the output is a 3-bit unsigned binary number *XYZ* which is the quotient of *ABCD* / 3. For example, if *ABCD* = 1100 (or 12 in decimal), then *XYZ* = 100 (or 4 in decimal); if *ABCD* = 0111 (or 7 in decimal), then *XYZ* = 010 (or 2 in decimal).

   (a) Draw the truth table and try to obtain the simplified SOP expressions of *X*, *Y*, and *Z* just from observation. (It is quite easy to obtain the simplified SOP expressions for *X* and *Y* just from observing the truth table, but harder for *Z*. The K-map technique is probably useful here, but we'll do that in the next tutorial.)

   (b) Verify your answers by first writing out their sum-of-minterms expressions and then simplifying the expressions from there using Boolean algebra. Write out the theorem(s) you use at each step.

   (c) From the simplified SOP expressions, implement *X*, *Y*, and *Z* using (i) 2-level AND-OR circuits, and (ii) 2-level NAND-only circuits. Assume that primed literals are not available.

Every question can be answered given enough time. However, you are to train yourself to seek the best method, the fastest way to solve a problem. Practice makes perfect!)